



Bro Communication

Robin Sommer

*Lawrence Berkeley National Laboratory &
International Computer Science Institute*

`robin@icir.org`

`http://www.icir.org`

Bro Workshop 2007



Overview

- Independent Bro instances can exchange information
- Basic communication model is *transparency*
 - Instances do not differentiate between own and remote state
 - E.g., events can be raised locally and/or received from peers
- Topics
 - Setting up communication
 - Exchanging events
 - Synchronizing variables
 - Coupling Bro with external applications



Setting Up Bro Communication

- Need to setup connections between all Bro instances
 - Fully meshed setup required (alternatives discussed later)
- Each node adds entries to `Remote::destinations`

```
@load remote
```

```
redef Remote::destinations += {  
  ["peer-1"] = [$host = 192.168.1.1, $connect=T, $retry=1min];  
  ["peer-2"] = [$host = 192.168.1.2, $connect=F];  
};
```

- `connect` defines which sides initiates the connection
 - `retry` triggers reconnect after given time if connection breaks down
- Nodes accepting connections load
 - `listen-clear.bro` for unsecured connections
 - `listen-ssl.bro` for SSL encrypted/authenticated connections
 - SSL needs certificate setup



Exchanging Events

- Each host specifies the events it wants to subscribe to

```
redef Remote::destinations += {  
  ["peer-1"] = [ ..., $events=/connection_established/ ];  
};
```

- Peer-1 will forward all `connection_established` events to us
- Receiver will execute handlers just as for local events
- Event subscriptions are given as a regular expression
- To subscribe to more events, use `/event1|event2|event3/`



Synchronizing Variables

- All script variables can be shared between peers
 - Each modification one instances does, is propagated to peers
 - Gives a global view of the variable's value
- To activate synchronization
 - `redef destinations += { ["peer-1"] = [..., $sync=T] };`
 - Choose variables to be shared:
`global my_set:set[addr] &synchronized;`
- Be careful of *loose synchronization* semantics
 - Synchronization is best-effort; no locking & no guarantees



Fine Tuning Communication

- Further communication capabilities
 - SSL for encryption & authentication
 - Optional data compression
 - Exchange of BPF capture filters
 - Hand-over for checkpointing
 - Proxy setup to avoid full meshes (routing in the future)
 - Aggregating logs centrally
- See `Remote::Destination` for more options
- Communication system provides the infrastructure for the *Bro cluster*.



Injecting External Information

- Events can be sent from external applications
- **Broccoli: The Bro Client Communications Library**
 - Open-source C library which speaks the Bro communication protocol
 - Easy to use in applications to send arbitrary events to a running Bro
 - We have used it to instrument, e.g., syslog, sshd, Apache
 - Script interface planned (Ruby, Python)
 - See `aux/broccoli/test/broping.c` for a simple example application
- **The *Generic Client* turns ASCII data into Bro events**
 - Based on Broccoli



The Generic Client

- Input for the generic client are lines of the form

```
event_name type=arg1 type=arg2 [...]
```

- This data is then piped into the client

```
> broclient host=<Bro-IP> <input.dat
```

- Example to send events

```
test_event(s: string, c: count)
```

```
> cat example.dat
```

```
test_event string=Foo count=42
```

```
test_event string=Bar count=43
```

```
> broclient host=192.168.1.1 <example.dat
```



Thanks for your attention.

Robin Sommer

*Lawrence Berkeley National Laboratory &
International Computer Science Institute*

`robin@icir.org`

`http://www.icir.org`

This work is supported by the Office of Science and Technology at the Department of Homeland Security. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security or the Office of Science and Technology.

